09/137,907

```
     FILE 'USPATFULL' ENTERED AT 15:33:09 ON 09 AUG 2001
L1             0 SEA CLIENT# INVOK? (P) SERVER# OBJECT? (P)(FAULT? OR ERROR?)
L2             5 SEA CLIENT# INVOK? (P) SERVER# (P)(FAULT? OR ERROR?)
                 D KWIC 1-5
                 D 1-5 FP
L3             0 SEA DETERMIN? (P) SERVER# (P)RELIABILY
L4           760 SEA SOFTWARE (P) FAULT? TOLERAN?
L5           372 SEA SERVER# OBJECT?
L6             7 SEA L4 AND L5
                 D 1-7 FP
                 D KWIC 6,7


     FILE HOME

     FILE USPATFULL
     FILE COVERS 1971 TO PATENT PUBLICATION DATE: 9 Aug 2001 (20010809/PD)
     FILE LAST UPDATED: 9 Aug 2001 (20010809/ED)
     HIGHEST GRANTED PATENT NUMBER: US6249914
     HIGHEST APPLICATION PUBLICATION NUMBER: US2001013137
     CA INDEXING IS CURRENT THROUGH 9 Aug 2001 (20010809/UPCA)
     ISSUE CLASS FIELDS (/INCL) CURRENT THROUGH: 9 Aug 2001 (20010809/PD)
     REVISED CLASS FIELDS (/NCL) LAST RELOADED: Apr 2001
     USPTO MANUAL OF CLASSIFICATIONS THESAURUS ISSUE DATE: Apr 2001

     >>> Page images are available for patents from 1/1/1998.  Patents   <<<
     >>> and applications are typically loaded on the day of publication.<<<
     >>> Page images are available for display by the following day.     <<<
     >>> Image data for the /FA field are available the following update.<<<

     >>> Complete CA file indexing for chemical patents (or equivalents) <<<
     >>> is included in file records.  A thesaurus is available for the  <<<
     >>> USPTO Manual of Classifications in the /NCL, /INCL, and /RPCL   <<<
     >>> fields.  This thesaurus includes catchword terms from the       <<<
     >>> USPTO/MOC subject headings and subheadings.  Thesauri are also  <<<
     >>> available for the WIPO International Patent Classification      <<<
     >>> (IPC) Manuals, editions 1-6, in the /IC1, /IC2, /IC3, /IC4,     <<<
     >>> /IC5, and /IC (/IC6) fields, respectively.  The thesauri in     <<<
     >>> the /IC5 and /IC fields include the corresponding catchword     <<<
     >>> terms from the IPC subject headings and subheadings.           <<<

     This file contains CAS Registry Numbers for easy and accurate
     substance identification.
```

descriptions of **server objects** registered for site global service from nodes at geographically remote sites upon node initialization or status change; receiving and registering said **server object** descriptions of global service registered **server objects** from other nodes at the local site; receiving and registering said **server object** descriptions of site global service registered **server objects** from nodes at geographically remote sites over the telecommunications network; storing said received **server object** descriptions in said **server object** database; receiving service requests from client objects, searching said **server object** database for a destination **server object** capable of performing said service requests, formatting said service requests into messages, and forwarding said messages to said destination **server object** at the local node, local site, or a geographically remote site.

14. The method, as set forth in claim 13, further comprising the steps of: storing a revision count in said **server object** database in response to the number of revisions made to said **server object** database; storing a local historical cache of past service requests and destination **server objects** capable of performing the service request in each process; storing a local revision count in said local historical cache; accessing. . . cache in response to a service request and said local revision count being equal to said revision count in said **server object** database; and accessing said **server object** database in response to a service request and said local revision count not being equal to said revision count in said **server object** database.

15. The method, as set forth in claim 14, wherein said **server object** description storing step further comprises the step of storing a node service state at object registration time for each registered. . .

. . . in claim 15, further comprising the step of: downgrading a node to a lower service state; and purging from said **server object** database registered objects resident in said downgraded node having a higher service state than said lower service state of said. . .

. . . site and at geographically remote sites of said upgraded node's higher service state; requesting for a copy of contents of **server object** database of said upgraded node; and storing said copy in **server object** database in all other nodes.

19. The method, as set forth in claim 13, wherein the step of storing **server object** description further comprises the step of storing an operation mode of said registered objects, said operation modes including active/standby and. . .

20. The method, as set forth in claim 13, wherein the step of storing **server object** description further comprises the step of storing a message queue ID for each registered object.

21. The method, as set forth in claim 13, wherein the step of storing **server object** description further comprises the step of storing a well known port ID for each registered object.

22. The method, as set forth in claim 13, wherein the step of storing **server object** description further comprises the step of storing an operation mode specifying an active/standby mode or load-sharing mode.

23. The method, as set forth in claim 22, wherein said step of searching

said **server object** database for destination **server objects** comprises the step of distributing service requests substantially equally among said destination **server objects** in response to said **server objects** being registered for load-sharing operating mode.

24. The method, as set forth in claim 22, wherein said step of searching said **server object** database for destination **server objects** comprises the step of distributing service requests substantially equally among said destination **server objects** using a round robin algorithm in response to said **server objects** being registered for load-sharing operating mode.

25. The method, as set forth in claim 13, wherein the step of forwarding said messages to said destination **server objects** comprises the step of establishing a communications link to a destination node in which said destination **server object** resides.

26. The method, as set forth in claim 13, wherein the step of forwarding said messages to said destination **server objects** comprises the step of establishing a TCP/IP communications link to a destination node in which said destination **server object** resides.

27. The method, as set forth in claim 13, wherein the step of forwarding said messages to said destination **server objects** comprises the step of forwarding said message via a UDP/TLI socket connection of a destination node in which said destination **server object** resides.

28. The method, as set forth in claim 25, wherein the step of forwarding said messages to said destination **server objects** comprises the steps of: forwarding said formatted message to a message handler process, said message handler process sending said message to a LinkTCP process in said destination node; and forwarding said message to said destination **server object**.

. . . 29. The method, as set forth in claim 13, further comprising the step of receiving a reply from said destination **server object**.

. . . plurality of processes executing therein, the method comprising the steps of: registering at least one instance of a plurality of **server objects** by a plurality of processes in each node, said **server objects** being registered for service for a local node, global service for a local site, and/or site global service for geographically. . . service object providing a representation of a portion of hardware and/or software services provided by the telephony applications; storing a **server object** description of each registered object in a **server object** database, said **server object** description includes an object name, a node name, a site name, and an instance name if more than one instance of an object is registered in the same node; broadcasting said **server object** descriptions of **server objects** registered for global service to other nodes at the local site upon registration; providing said **server object** descriptions of **server objects** registered for site global service to nodes at

geographically remote sites over a telecommunications network upon request therefrom at node initialization or status change, said broadcasted and provided **server object** descriptions being stored in **server object** databases at other nodes at the local site and nodes of geographically remote sites respectively; requesting **server object** descriptions of **server objects** registered for site global service from nodes at geographically remote sites over the telecommunications network upon node initialization or status change; receiving and registering said **server object** descriptions of registered **server objects** from other nodes at the local site and from nodes of geographically remote sites over the telecommunications network; storing said received **server object** descriptions in said **server object** database; receiving service requests from client objects, searching said **server object** database for destination **server object** capable of performing said service requests, formatting said service requests into messages, and forwarding said messages to said destination **server objects** at the local site or a geographically remote site.

31. The method, as set forth in claim 30, further comprising the steps of: storing a revision count in said **server object** database in response to the number of revisions made to said **server object** database; storing a local historical cache of past service requests and destination **server objects** capable of performing the service request in each process; storing a local revision count in said local historical cache; accessing. . . cache in response to a service request and said local revision count being equal to said revision count in said **server object** database; and accessing said **server object** database in response to a service request and said local revision count not being equal to said revision count in said **server object** database.

32. The method, as set forth in claim 31, wherein said **server object** description storing step further comprises the step of storing a node service state at object registration time for each registered. . .

. . . in claim 32, further comprising the step of: downgrading a node to a lower service state; and purging from said **server object** database registered objects resident in said downgraded node having a higher service state than said lower service state of said. . .

. . . site and at geographically remote sites of said upgraded node's higher service state; requesting for a copy of contents of **server object** database of said upgraded node; and storing said copy in **server object** database in all other nodes.

36. The method, as set forth in claim 30, wherein the step of storing **server object** description further comprises the step of storing an operation mode of said registered objects, said operation modes including active/standby and. . .

37. The method, as set forth in claim 30, wherein the step of storing **server object** description further comprises the step of storing a message queue ID for each registered object.

38. The method, as set forth in claim 30, wherein the step of storing **server object** description further comprises the step of storing a well known port ID for each registered object.

39. The method, as set forth in claim 30, wherein the step of storing **server object** description further comprises the step

of storing an operation mode specifying an active/standby mode or load-sharing mode.

40. The method, as set forth in claim 39, wherein said step of searching said **server object** database for destination **server objects** comprises the step of distributing service requests substantially equally among said destination **server objects** in response to said **server objects** being registered for load-sharing operating mode.

41. The method, as set forth in claim 39, wherein said step of searching said **server object** database for destination **server objects** comprises the step of distributing service requests substantially equally among said destination **server objects** using a round robin algorithm in response to said **server objects** being registered for load-sharing operating mode.

42. The method, as set forth in claim 30, wherein the step of forwarding said messages to said destination **server objects** comprises the step of establishing a communications link to a destination node in which said destination **server object** resides.

43. The method, as set forth in claim 30, wherein the step of forwarding said messages to said destination **server objects** comprises the step of establishing a TCP/IP communications link to a destination node in which said destination **server object** resides.

44. The method, as set forth in claim 30, wherein the step of forwarding said messages to said destination **server objects** comprises the step of forwarding said message via a UDP/TLI socket connection of a destination node in which said destination **server object** resides.

45. The method, as set forth in claim 42, wherein the step of forwarding said messages to said destination **server objects** comprises the steps of: forwarding said formatted message to a message handler process, said message handler process sending said message to a LinkTCP process in said destination node; and forwarding said message to said destination **server object**.

. . . 46. The method, as set forth in claim 30, further comprising the step of receiving a reply from said destination **server object**.

L6 ANSWER 6 OF 7 USPATFULL
United States Patent Patent Number: 5905860
Date of Patent: 18 May 1999

Fault tolerant electronic licensing system

Inventor(s): Olsen, James E., Park City, UT, United States
Bringhurst, Adam L., Provo, UT, United States
Assignee: Novell, Inc., Provo, UT, United States (U.S. corporation)
Appl. No.: 97-804936
Filed: 24 Feb 1997

Publication Details

PATENT INFORMATION: US 5905860 18 May 1999

Related U.S. Application Data

Continuation-in-part of Ser. No. US 1996-620319, filed on 15 Mar 1996, now patented, Pat. No. US 5758069, Pat. No. 5758069

Int. Cl. H04L009-00
Issue U.S. Cl. 395/187.010; 395/186.000; 395/200.570; 380/004.000
Current U.S. Cl. 713/201.000; 705/059.000; 709/227.000; 713/200.000
Field of Search 395/187.01; 395/186; 395/200.57; 380/3; 380/4

Reference Cited

PATENT DOCUMENTS

Patent Number	Date	Class	Inventor
US 4870568	Sep 1989	707/005.000	Kahle et al.
US 4924378	May 1990	395/187.010	Hershey
US 4937863	Jun 1990	380/004.000	Robert et al.
US 4941175	Jul 1990	380/004.000	Enescu et al.
US 5023907	Jun 1991	380/004.000	Johnson et al.
US 5047928	Sep 1991	380/004.000	Wiedemer
US 5097533	Mar 1992	395/500.000	Burger et al.
US 5103476	Apr 1992	380/004.000	Waite et al.
US 5138712	Aug 1992	395/186.000	Corbin
US 5157663	Oct 1992	380/025.000	Major et al.
US 5182770	Jan 1993	380/004.000	Medveczky et al.
US 5187770	Feb 1993	385/145.000	Mishima et al.
US 5201046	Apr 1993	707/100.000	Goldberg et al.
US 5201048	Apr 1993	707/003.000	Coulter et al.
US 5204897	Apr 1993	380/004.000	Wyman
US 5222134	Jun 1993	380/004.000	Waite et al.
US 5260999	Nov 1993	380/004.000	Wyman
US 5265065	Nov 1993	707/004.000	Turtle
US 5278980	Jan 1994	707/004.000	Pederson et al.
US 5287408	Feb 1994	380/025.000	Samson
US 5319705	Jun 1994	380/004.000	Halter et al.
US 5343526	Aug 1994	380/004.000	Lassers
US 5343527	Aug 1994	380/004.000	Moore
US 5349642	Sep 1994	380/025.000	Kingdon

US 5371792	Dec 1994	380/003.000	Asai et al.
US 5375206	Dec 1994	395/712.000	Hunter et al.
US 5386369	Jan 1995	705/400.000	Christiano
US 5438508	Aug 1995	705/008.000	Wyman
US 5553143	Sep 1996	380/025.000	Ross
US 5675782	Oct 1997	395/609.000	Montague et al.
US 5689638	Nov 1997	395/188.010	Sadovsky
AU 3820993	Apr 1993		
AU 4811393	Mar 1994		
WO 8904520	May 1989		
WO 9220021	Nov 1992		

Art Unit - 275
Primary Examiner - Hua, Ly
Attorney, Agent or Firm - Dinsmore & Shohl LLP

---

22 Claim(s), 10 Drawing Figure(s), 5 Drawing Page(s)

ABSTRACT

A licensing system provides enhanced flexibility for licensing applications in a network. The licensing system includes a directory services database which stores all license information. The directory services database is accessed by providing a request to a license service provider associated with a server. The
license service provider generates an executable entity based on the request parameters, which searches the database and, if the appropriate units are available, assembles a license. The license and the application are then transmitted to the requesting client. All aspects of the transaction are also stored in a database organized according to a transaction's relation to a particular license.

L6 ANSWER 7 OF 7 USPATFULL
United States Patent Patent Number: 5892946
Date of Patent: 6 Apr 1999

System and method for multi-site distributed object management environment

Inventor(s): Woster, George W., Dallas, TX, United States
Linares, Melissa A., Plano, TX, United States
Shah, Mahesh V., Plano, TX, United States
Assignee: Alcatel USA, Inc., Plano, TX, United States (U.S. corporation)
Appl. No.: 95-526953
Filed: 12 Sep 1995

Publication Details

PATENT INFORMATION: US 5892946 6 Apr 1999

Int. Cl. G06F009-40
Issue U.S. Cl. 395/680.000
Current U.S. Cl. 709/316.000
Field of Search 395/683; 395/614; 395/680; 395/200.47; 395/200.48; 395/200.49

Reference Cited

PATENT DOCUMENTS

Patent Number	Date	Class	Inventor

US 5287507	Feb 1994	395/683.000	Hamilton et al.
US 5301319	Apr 1994	395/614.000	Thurman et al.
US 5303375	Apr 1994	395/670.000	Collins et al.
US 5307490	Apr 1994	395/684.000	Davidson et al.
US 5377350	Dec 1994	395/683.000	Skinner
US 5396630	Mar 1995	395/683.000	Banda et al.
US 5410688	Apr 1995	395/610.000	Williams et al.
US 5522077	May 1996	395/683.000	Cuthbert et al.
US 5551035	Aug 1996	395/683.000	Arnold et al.
US 5596720	Jan 1997	395/200.030	Hamada et al.
US 5666479	Sep 1997	395/180.000	Kashimoto et al.
US 5734902	Mar 1998	395/683.000	Atkins et al.
US 5758078	May 1998	395/200.330	Kurita et al.
EP 474340	Aug 1990	G06F009-44	
EP 497022	Jan 1991	G06F015-40	
WO 91300772	Jan 1991	G06F015-40	
WO 91306128	Jul 1991	G06F009-44	

OTHER PUBLICATIONS

Robert Orfali and Dan Harkey, "Client/Server With Distributed Objects", Apr., 1995, BYTE, pp. 151-162.

Maruyama, Katsumi, "Object-Oriented Switching Software Technology," IEICE Transactions on Communications, vol. E75-B, No. 10, Oct. 1992.

Gentry, Dennis, "Spreading the Wealth: DO and PDO", NeXTSTEP Developer Journal,
p. (7), 1994.

Karve, Anita, "Operator, Give me Telephony", LAN Magazine, p. (7), Jul. 1994.

Orfali, Robert, Dan Harkey, "Client/Server Survival Guide with OS/2.RTM.", Van Nostrand Reinhold,, pp. 249-251, 402-409, 689-723, 757-771, 1994.

Art Unit - 275
Primary Examiner - Oberley, Alvin
Assistant Examiner - Courtenay, III, St. John
Attorney, Agent or Firm - Baker & Botts, L.L.P.

---

46 Claim(s), 13 Drawing Figure(s), 7 Drawing Page(s)

ABSTRACT

A distributed object messaging system and method (10) are provided for a plurality of nodes (15-21, 25-32) distributed in multiple physically separate sites (12, 13). There are a plurality of processes (40-44) executing in each node. The processes register a plurality of objects in each node. The objects include client objects and **server objects**. The
server **objects** may be registered for global service for
service availability to client objects in a local node, local site and/or site
global service for service availability for client object in remote sites. a
server **object** database (46) is used in each node to store a
server **object** description for each **server**
object registered in the node and objects registered in remote nodes
that are registered for global or site global service. a client-server
interface (50) is accessible by client objects and receives requests for
services therefrom. The client-server interface accesses the **server**
object database (46) for at least one destination **server**
object capable of performing the client object's requested service,
and
forwards the service request to the destination **server**
objects at a local site or a remote site.

=> d kwic 6,7

L6 ANSWER 6 OF 7 USPATFULL
DETD A licensing system according to various aspects of the present invention
provides a flexible, robust, and **fault tolerant**
licensing apparatus and method for licensing **software** in a
network. Referring now to FIGS. 1A and 1B, a licensing system 100
according to various aspects of the. . .
DETD . . . is configured to support a transactional database 105. The NCP
pointer facilitates location of and communication with the corresponding
NCP **server object**.

L6 ANSWER 7 OF 7 USPATFULL
AB . . . executing in each node. The processes register a plurality of
objects in each node. The objects include client objects and
server objects. The **server objects**
may be registered for global service for service availability to client
objects in a local node, local site and/or site global service for
service availability for client object in remote sites. a **server**
object database (46) is used in each node to store a
server object description for each **server**
object registered in the node and objects registered in remote
nodes that are registered for global or site global service. a
client-server interface (50) is accessible by client objects and
receives requests for services therefrom. The client-server interface
accesses the **server object** database (46) for at
least one destination **server object** capable of
performing the client object's requested service, and forwards the
service request to the destination **server objects** at
a local site or a remote site.
SUMM . . . from multiple servers at the same time. In additiona, there is
a requirement in telephony applications for both hardware and
software redundancy to provide a hardware/**software**
fault tolerant service environment. This environment
consists of nodes in geographically remote locations in the global
telecommunications network. The system must be. . .
SUMM . . . executing in each node. The processes register a plurality of
objects in each node. The objects include client objects and
server objects. The **server objects**
may be registered for global service for service availability to client
objects in a local node, local site and/or site global service for
service availability for client object in remote sites. A **server**
object database is used in each node to store a **server**
object description for each **server object**
registered in the node and objects registered in remote nodes that are
registered for global or site global service. A client-server interface
is accessible by client objects and receives requests for services
therefrom. The client-server interface accesses the **server**
object database for at least one destination **server**
object capable of performing the client object's requested
service, and forwards the service request to the destination
server objects at a local site or a remote site.
SUMM In another aspect of the invention, a local cache is kept in each
process to record past service requests and **server**
object matches. The local cache is accessed first for service
requests for a match if the **server object** database

has not been altered since the last database access.

SUMM In yet another aspect of the invention, the current service state of a node is recorded in the **server object** database whenever one of its objects is registered. The contents of the **server object** database can then be adjusted in response to a node's upgrade or downgrade in service status.

DRWD FIG. 5 is an exemplary **server object** database showing the type of information that is stored therein;

DRWD FIG. 7 is an exemplary flowchart of a get list of **server objects** process;

DRWD FIG. 8 is an exemplary flowchart of a request list of **server objects** process;

DRWD FIG. 9 is an exemplary flowchart of a receive list of **server objects** process;

DETD . . . addition, each node includes a DOME server process 48 which function as a registration interface between the client objects and **server objects** that reside in different processes within a node or in different nodes in the system, whether at the same site. . . register one or more instances of objects with DOME server process 48, which stores information of registered objects in a **server object** database 46 that is accessible to the processes 40-44 in the node and other nodes at the same or different sites. **Server object** database 46 is read accessed by a client DOME interface every time the client requests service from a **server object**. All local **server objects** are accessed directly by the DOME client interface by placing the service request directly on the **server object**'s message queue. A **server object** is defined as an encapsulation of functionality and data that are accessible to remote client objects by method invocations.

DETD . . . invoke the DOME server process 48 and register its objects and server member functions, as shown in block 52. A **server object** has one or more server member functions which have a unique user defined interface for receiving from and sending call. . . parameters to a client object. Server member functions are referenced by clients using identification numbers that are recorded in the **server object** descriptions. Each member function has a function prototype definition registered by the **server object**. A server member function description database (not shown) is maintained in each process to record the registered server member function descriptions of registered **server objects** in the process.

DETD After object registration, the DOME server process 48 then initializes **server object** database 46, as shown in block 54. In blocks 56 and 58, DOME server process 48 then requests the platform. . . object server database 46 with information about objects registered for global or site global service in other nodes. A global **server object** is visible to client objects in the same site while a site global **server object** is visible to client objects in other sites as well.

DETD Referring also to FIG. 5, **server object** database 46 contains object descriptions such as the object name, node name, site name, and instance name. The object, node,. . . following a predetermined naming convention. The node and/or site names may also be indices to respective lists kept in the **server object** database. Further, the names may be concatenated in a predetermined manner. Multiple copies of an object can be registered without. . .

DETD **Server object** database 46 further includes the node operating status at object registration time. Node operating status includes out of service, operating. . . object can service requests from client objects local to its resident node, site, or client objects from other sites, respectively. **Server object** database 46 further includes the communication options and specific

information therefor of the object, such as whether TCP/IP or User. . .

DETD . . . DOME server process 48 then requests that it be notified whenever any node changes its operating status, so that its **server object** database may be updated to reflect the changes. DOME server process 48 is then ready to provide service and waits. . .

DETD . . . registered in block 74. If the object is already registered, its message queue is removed and its corresponding entry in **server object** database 46 is deleted, as shown in blocks 76 and 78. Subsequently in block 80, the object description of the object instance is added to **server object** database 46.

DETD . . . at the time of registration. Site global objects are revealed to other sites when remote site nodes request for the **server object** database, which typically occurs on node status change.

DETD Referring to FIG. 6, a simplified flowchart for requesting service from a **server object** for a client object 90 is shown. The client object makes a service request and provides the calling parameters. DOME. . . server member function into a service request message, as shown in block 92. DOME then requests for a list of **server objects** matching the client's full or partial **server object** description, as shown in block 94. The **server object** database 46 (FIG. 2) is organized in such a way that a DOME process can readily access a particular object. . . to a list of node entries, and each node entry points to a list of named instances. DOME selects a **server object** from the list as the destination for the service request, sends the request service message to the selected **server object**, and exits, as shown in blocks 95-100.

DETD 2. If the client object does not specify a destination, the request is routed to the local site **server object**(s) using the active/standby and load-sharing routing rules.

DETD 3. If a site name or remote site specifier is used as a destination, only site global registered **server objects** in that site is considered as destination targets and the active/standby and load-sharing routing rules apply.

DETD **Server objects** and nodes may be registered as either active/standby or active (load-sharing) mode. Active objects are load-shared using a round robin. . .

DETD There are eight different **server object** routing algorithms which DOME may use when processing a client request for service. If a site name is specified in. . .

DETD When the client makes a service request, the object name is searched for in the **server object** database to determine if the object is an active/standby type or an always active type. Next, the node name is. . .

DETD Each of the eight **server object** routing algorithms consists of two parts. In the client node, DOME selects the best server node to service the request. . . node. This part of service routing is the server node selection method. In the server node, DOME selects the best **server object** instance to receive the request. This part of service routing is known as the server instance selection method. The above. . .

DETD 1. The client specifies a unique node name which must process the service request. First, the **server object** database is accessed to determine if the **server object** resides in the node name specified. If not found in the specified node, DOME declares an object not found error. If the server instance name is specified, then the **server object** database is searched to ensure that the instance named resides in the specified node name. An instance not found error. . .

DETD . . . The client DOME constructs an instance list of all instances in all nodes of the requested object name. The registered **server object** receive states is found in the **server object** database. The **server object** can be registered to receive in the active, standby, or active/standby state. The instance list is scanned for instances of. . .

DETD . . . to send the service request to a specific instance name which can exist in any node in the system. The **server object** database is checked to determine which node the instance name resides in.

DETD . . . client object may make a specialized request to a server in another node that is not listed in the local **server object** database. These are servers that have not been registered as site global **server objects**. This specialized request is forwarded to the Message Handler process, which looks up the destination queue ID for the requested object and forwards the message to that object's DOME server process. The DOME server interface of the **server object** then receives the service request message and format the parameters contained in the message into the input sequence expected by the **server object**. The **server object** then receives the message and performs the requested service.

DETD FIG. 7 provides some additional details of the process for getting the list of **server objects** 110. Referring also to FIG. 2, **server object** database 46 includes a revision counter 111, which is incremented every time the contents of the database is modified. Each. . . has a local cache 112, the contents of which is a historical record of past service requests and corresponding matched **server object** to perform the service requested. Local cache 112 also keeps a record of what the revision counter count was at. . . read and compared with the local record of the count. If the numbers do not match, then the contents of **server object** database 46 have been altered and the contents of local cache 112 are no longer valid. Therefore, **server object** database 46 is accessed by first locking the semaphore, reading the relevant entry or entries in database 46, and then. . . be searched for a match, a shown in block 124. Also, if there is no match with the service request, **server object** database 46 is accessed in a similar manner as shown in blocks 118-122. Subsequently, the process ends in block 126.

DETD . . . first initialized, it requests other nodes at the same site to supply it with the object descriptions stored in their **server object** databases, as shown in block 64 of FIG. 3. Site global servers are not broadcast to remote sites as they are registered; site global **server objects** become visible to remote site clients when the remote site nodes ask for the **server object** database, which typically occurs on node status change. The request for list of **server objects** process 130 is shown in FIG. 8. A determination is made as to whether the request is from a node. . . site or a remote site in block 132. If the request is from a local site, the object descriptions of **server objects** registered for global service is obtained, as in block 134. Else, the object description desired is of **server objects** registered for site global service, which is obtained in block 136. The object description is then communicated to the requesting. . .

DETD . . . a node at the local site or a remote site 150. Upon receipt of the object description, the semaphore of **server object** database is obtained to lock out other processes, as shown in block 152.

In block 154, the revision counter kept for the database is then incremented to reflect that the database is modified. The **server**

object description for the node in question is then deleted and the updated description is inserted into the **server object** database, as shown in blocks 156 and 158. The semaphore is then released in block 160 and the process ends. . .

DETD . . . Message Handler and LinkTCP processes. FIG. 10 illustrates the communications process. FIG. 10 shows a client object 174 and a **server object** 176, where client and **server objects** reside in different nodes X and Y, 170 and 172 respectively. When **server object** 172 resides in a different node, DOME uses a network logical link interface provided by the two specialized processes, the. . . Message Handler then forwards the request to the destination node LinkTCP process 180, which places the service request in the **server object** IPC system V message queue 182 to be received by **server object** 176. **Server object** may be informed of the service request by a predetermined signal announcing the arrival of the message. Alternatively, **server object** may poll for the existence of service requests or suspend until a service request is present. Each message queue 182. . .

DETD . . . to the client with the server reply. The client object may then be assured that the correct reply from the **server object** is returned for the request that was made. Note that when the destination **server object** resides in the same node as the client object, the service request message is transmitted directly from client object to the message queue of the **server object**.

DETD . . . inter-node communications. A client object 174 residing in node X 170 may send a service request via UDP to a **server object** 176 residing in node Y 172. Results of the service performed, if any, may be transmitted back to client object. . . same manner. This method involves less overhead than the TCP/IP transmission method but does not guarantee message delivery. A global **server object** may broadcast its well-known UDP port ID to other nodes where it is registered. A site global **server object** 's well-known UDP port ID is made available to nodes in other sites when the **server object** database of its resident node is requested by the remote nodes.

DETD Referring to FIG. 12, a set node status process 190 is used to automatically update a **server object** database to reflect a node status change. Every time the status or service state of a remote node is changed, the DOME server 48 (FIG. 2) process of a node updates its **server object** database. The node receives the new status of the remote node that experienced a change in status, and sets the. . . in blocks 194 and 196, indicating that the node is no longer up and running, then the contents of the **server object** database must be updated to remove the registered objects of the downgraded node. This is done by first locking the semaphore of the **server object** database and incrementing the revision counter, as shown in blocks 198 and 200. The registered **server objects** resident in the downgraded node are then deleted from the **server object** database, as shown in block 202. The semaphore is then released in block 204. The **server object** database is therefore purged of all registered objects that were resident in the out of service node.

DETD . . . as determined in blocks 194 and 210, or the new status is in service, as determined in block 216, the **server object**

database may need to be updated depending on whether the node status is upgraded or downgraded, as determined in block. . .

DETD . . . block 218 be true. The boolean variable UPDATE may be used to positively request that the database be purged of **server objects** that are no longer available for service. If the condition in block 218 is false, then a copy of the entire **server object** database of the remote node is requested and added to the local database, as shown in block 220. The **server object** request and receive processes are shown in FIGS. 8 and 9 described above.

DETD If the condition in block 218 is true, then the contents of the **server object** database of the local node need to be selectively deleted. If there is a downgrade in service state, then the.

. . service state higher than the current new status are no longer valid and their respective description is removed from the **server object** database. The semaphore of the **server object** database is first secured, as shown in block 222. The revision counter is then incremented to reflect this recent change. . . block 228, and the procedure ends in block 230. Because the service state of the node is noted in the **server object** database when an object registers, the database may be automatically updated to remove those objects that are no longer valid. . .

CLM What is claimed is:

. . . of processes perform telephony applications; said plurality of processes registering a plurality of objects, said objects including client objects and **server objects**, and each object having at least one instance thereof, wherein each object is a representation of a portion of hardware and software services provided by the telephony applications; said plurality of **server objects** being selectively registered for global service for service availability to client objects in a local node and a local site,. . . global service for service availability to client objects in a processor node at geographically remote sites, each processing node

registering **server objects** from other nodes at its local site, and nodes from geographically remote sites, which have been registered for global, or site global, service, wherein **server objects** registered for global service are broadcast to other processor nodes at the local site upon registration and **server objects** registered for site global service are sent to a particular processor node at a remote site over the telecommunications network upon a request therefrom and not broadcast upon registration; a **server object** database residing in each processor node storing a **server object** description for each registered **server object**, said **server object** description including an object name, a node name, a site name, and an instance name if more than one instance. . . processor node; and a client-server interface being accessible by said client objects and receiving requests for services therefrom, accessing said **server object** database for at least one destination **server objects** capable of performing said requested service, formatting said service request into a message, and forwarding said message to said destination **server objects** at a local node, load site, or a processor node at a geographically remote site.

. . . The system, as set forth in claim 1, wherein said plurality of processes further comprises a local cache of past **server object** database access requests and destination **server objects**.

3. The system, as set forth in claim 2, wherein said **server object** database further comprises a revision counter recording a

count of revisions performed on said **server object** database, said client-server interface accessing said local cache for matching **server objects** in response to a local revision count being equal to said count of said revision counter in said **server object** database, and accessing said **server object** database in response to said local revision count being unequal to said count of said revision counter in said **server object** database.

4. The system, as set forth in claim 1, further comprising: a message handler process residing in each processor node for sending messages to destination **server objects** residing in processor nodes at geographically remote sites; and a LinkTCP process residing in each processor node for receiving messages. . .

6. The system, as set forth in claim 1, wherein said **server object** description stored in said **server object** database further includes a processor node service state at object registration time.

8. The system, as set forth in claim 6, wherein said **server object** database is purged of registered objects having a higher service state than a current service state of a resident node. . .

9. The system, as set forth in claim 6, wherein said **server object** database is replaced by a new copy of a **server object** database of a processor node at a geographically remote site in response to an upgrade of service state of said. . .

10. The system, as set forth in claim 1, wherein said **server object** description stored in said **server object** database further includes an operation mode of said registered objects, said operation modes including active/standby and load-sharing modes.

11. The system, as set forth in claim 1, wherein said **server object** description stored in said **server object** database further includes a message queue ID for each registered object.

12. The system, as set forth in claim 1, wherein said **server object** description stored in said **server object** database further includes a well known port ID for each registered object.

. . . plurality of processes executing therein, the method comprising the steps of: registering at least one instance of a plurality of **server objects** by a plurality of processes in each node, said **server objects** being registered for local, global, and/or site global service, wherein each of the plurality of processes perform telephony applications, each **server object** providing a representation of a portion of hardware and/or software services provided by the telephony applications; storing a **server object** description of each registered object in a **server object** database, said **server object** description includes an object name, a node name, a site name, and an instance name if more than one instance of an object is registered in the same node; broadcasting said **server object** descriptions of **server objects** registered for global service to other nodes at a local site upon registration; providing said **server object** descriptions of **server objects** registered for site global service to nodes at geographically remote sites over a telecommunications network upon request therefrom at node initialization or status change; requesting **server object**

L2 ANSWER 1 OF 5 USPATFULL
United States Patent Patent Number: 2001000812 A1
Date of Patent: 3 May 2001

Leasing for failure detection

Inventor(s): Waldo, James H., Dracut, MA, United States
Wollrath, Ann M., Groton, MA, United States
Scheifler, Robert, Somerville, MA, United States
Arnold, Kenneth C.R.C., Lexington, MA, United States
Assignee: Sun Microsystems, Inc. (U.S. corporation)
Appl. No.: 2000-739744 A1
Filed: 20 Dec 2000

Publication Details

PATENT INFORMATION: US 2001000812 A1 3 May 2001

Related U.S. Application Data

Continuation of Ser. No. US 1999-377491, filed on 20 Aug 1999, PENDING
Division of Ser. No. US 1998-44916, filed on 20 Mar 1998, GRANTED, Pat. No. US 6016500, Pat. No. 6016500
Continuation-in-part of Ser. No. US 1996-729421, filed on 11 Oct 1996, GRANTED, Pat. No. US 5832529, Pat. No. 5832529

Priority Data

L2 5 CLIENT# INVOK? (P) SERVER# (P)(FAULT? OR ERROR?)

=> d kwic 1-5

L2 ANSWER 1 OF 5 USPATFULL
DETD 154. The **client invokes** the "cancel" method when the client wishes to cancel the lease. Thus, invocation of the cancel method allows the **server** to re-claim the storage locations so that other programs may access them. Accordingly, the cancel method ensures that the **server** can optimize the use of the storage locations in the distributed system. It should be noted that if the lease expires without an explicit cancellation by the client, the **server** assumes an **error** occurred.
DETD . . . continues to step 10005. However, if the lease is about to expire, the client sends a renew request to the **server** (step 10009). In this step, the **client invokes** the renew method on the lease object. After invoking the renew method, the client determines if the renew request was. . . method returned successfully. If so, processing continues to step 10005. However, if the renew method did not complete successfully, the **client invokes** the recover method on the lease object (step 10012). Because the renew request did not complete successfully, the client knows that a failure occurred and thus needs to perform **error** recovery by invoking the recover method. The recover method then performs recovery on the **server**.

L2 ANSWER 2 OF 5 USPATFULL
DETD The **client invokes** the "cancel" method when the client wishes to cancel the lease. Thus, invocation of the cancel method allows the **server** to re-claim the storage locations so that other programs may access them. Accordingly, the cancel method ensures that the **server** can optimize the use of the storage locations in the distributed system. It should be noted that if the lease expires without an explicit cancellation by the client, the **server** assumes an **error** occurred.
DETD . . . continues to step 10005. However, if the lease is about to expire, the client sends a renew request to the **server** (step 10009). In this step, the **client invokes** the renew method on the lease object. After invoking the renew method, the client determines if the renew request was. . . method returned successfully. If so, processing continues to step 10005. However, if the renew method did not complete successfully, the **client invokes** the recover method on the lease object (step 10012). Because the renew request did not complete successfully, the client knows that a failure occurred and thus needs to perform **error** recovery by invoking the recover method. The recover method then performs recovery on the **server**.

L2 ANSWER 3 OF 5 USPATFULL
DETD While the **server** 10 performs the cleanup operation, the client 12 checks the data packet and displays an **error** message stating that the **server** 10 could not communicate with the scanner and prompts the user to determine if the user wishes to select another **server** 10 (block 67). If the user so indicates, the

client invokes a process to select another **server** 10. The preferred process is the discovery method mentioned above (block 20).

L2 ANSWER 4 OF 5 USPATFULL

DETD The **client invokes** the "cancel" method when the client wishes to cancel the lease. Thus, invocation of the cancel method allows the **server** to re-claim the storage locations so that other programs may access them. Accordingly, the cancel method ensures that the **server** can optimize the use of the storage locations in the distributed system. It should be noted that if the lease expires without an explicit cancellation by the client, the **server** assumes an **error** occurred.

DETD . . . continues to step 10005. However, if the lease is about to expire, the client sends a renew request to the **server** (step 10009). In this step, the **client invokes** the renew method on the lease object. After invoking the renew method, the client determines if the renew request was. . . method returned successfully. If so, processing continues to step 10005. However, if the renew method did not complete successfully, the **client invokes** the recover method on the lease object (step 10012). Because the renew request did not complete successfully, the client knows that a failure occurred and thus needs to perform **error** recovery by invoking the recover method. The recover method then performs recovery on the **server**.

L2 ANSWER 5 OF 5 USPATFULL

AB An **error** recovery technique is provided which is used during the automatic installation of software on a client data processing system from a connected **server** data processing system. The installation process is made up of a plurality of operations executing on the client, for example. . . operations on the client is watchdog timer code which preferably takes the form of device driver software received from the **server** system early in the installation process. At the beginning of each operation, a preselected time value is written to a. . . normally taken for the operation to complete. If the counter expires before the operation is complete (if for example the **server** system or network fails), the **client invokes** an **error** recovery procedure which may for example involve rebooting the client.

=> d 1-5 fp

L2 ANSWER 1 OF 5 USPATFULL

United States Patent Patent Number: 2001000812 A1
Date of Patent: 3 May 2001

Leasing for failure detection

Inventor(s): Waldo, James H., Dracut, MA, United States
Wollrath, Ann M., Groton, MA, United States
Scheifler, Robert, Somerville, MA, United States
Arnold, Kenneth C.R.C., Lexington, MA, United States
Assignee: Sun Microsystems, Inc. (U.S. corporation)
Appl. No.: 2000-739744 A1
Filed: 20 Dec 2000

Publication Details

PATENT INFORMATION: US 2001000812 A1 3 May 2001

Related U.S. Application Data

Continuation of Ser. No. US 1999-377491, filed on 20 Aug 1999, PENDING
Division of Ser. No. US 1998-44916, filed on 20 Mar 1998, GRANTED, Pat. No. US 6016500, Pat. No. 6016500
Continuation-in-part of Ser. No. US 1996-729421, filed on 11 Oct 1996, GRANTED, Pat. No. US 5832529, Pat. No. 5832529

Priority Data

US 1998-76048 26 Feb 1998 (60)

Int. Cl. G06F015-173; G06F015-16
Issue U.S. Cl. 709/225.000; 709/224.000; 714/015.000; 709/229.000; 713/201.000
Current U.S. Cl. 709/225.000; 709/224.000; 714/015.000; 709/229.000; 713/201.000
Attorney, Agent or Firm - FINNEGAN, HENDERSON, FARABOW, GARRETT &, DUNNER LLP,
1300 I STREET, NW, WASHINGTON, DC, 20005

---

3 Claim(s)11 Drawing Page(s)

ABSTRACT

A system for using a lease to detect a failure and to perform tailure recovery is provided in using this system, a client requests a lease from a server to utilize a resource managed by the server for a period of time. Responsive to the request, the server grants the lease, and the client continually requests renewal of the lease. If the client fails to renew the lease, the server detects that an error has occurred to the client. Similarly, if the server fails to respond to a renew request, the client detects that an error has occurred to the server. As part of the lease establishment, the client and server exchange failure-recovery routines that each invokes if the other experiences a failure.

L2 ANSWER 2 OF 5 USPATFULL
United States Patent Patent Number: 6243716 B1
Date of Patent: 5 Jun 2001

Methods and systems for distributed failure detection and recovery using leasing

Inventor(s): Waldo, James H., Dracut, MA, United States
Wollrath, Ann M., Groton, MA, United States
Scheifler, Robert, Somerville, MA, United States
Arnold, Kenneth C. R. C., Lexington, MA, United States
Assignee: Sun Microsystems, Inc., Palo Alto, CA, United States (U.S. corporation)
Appl. No.: 99-377491
Filed: 20 Aug 1999

Publication Details

PATENT INFORMATION: US 6243716 B1 5 Jun 2001

Related U.S. Application Data

Division of Ser. No. US 1998-44916, filed on 20 Mar 1998, now patented, Pat. No. US 6016500, Pat. No. 6016500
Continuation-in-part of Ser. No. US 1996-729421, filed on 11 Oct 1996, now patented, Pat. No. US 5832529, Pat. No. 5832529

Priority Data

US 1998-76048 26 Feb 1998 (60)

Int. Cl. G06F015-00
Issue U.S. Cl. 707/202.000; 707/010.000; 707/206.000; 709/203.000; 709/219.000; 709/224.000; 709/229.000
Current U.S. Cl. 707/202.000; 707/010.000; 707/206.000; 709/203.000; 709/219.000; 709/224.000; 709/229.000
Field of Search 707/10; 707/202; 707/206; 707/204; 707/9; 709/203; 709/219; 709/224; 709/229

Reference Cited

PATENT DOCUMENTS

L14 ANSWER 1 OF 4 USPATFULL
AN 2001:76645 USPATFULL
TI Duplicated naming service in a distributed processing system
IN Jacobs, Dean B., Berkeley, CA, United States
Halpern, Eric M., San Francisco, CA, United States
PA BEA Systems, Inc., San Jose, CA, United States (U.S. corporation)
PI US 6236999 B1 20010522
AI US 1999-405508 19990923 (9)
PRAI US 1998-107167 19981105 (60)
DT Utility
FS Granted
LN.CNT 1079
INCL INCLM: 707/010.000
INCLS: 707/100.000; 707/104.000; 709/200.000; 709/242.000
NCL NCLM: 707/010.000
NCLS: 707/100.000; 709/200.000; 709/242.000
IC [7]
ICM: G06F017-30
EXF 707/10; 707/103; 707/104; 707/100; 709/242; 709/200

L14 ANSWER 2 OF 4 USPATFULL
AN 2001:69000 USPATFULL
TI Shared-everything file storage for clustered system
IN Edmonds, Paul, Palo Alto, CA, United States
Zhang, Yi, Sunnyvale, CA, United States
Xu, Chang, San Jose, CA, United States
Doshi, Priyen, San Jose, CA, United States
Co, Stephen, San Ramon, CA, United States
Tel, Michael P., Sunnyvale, CA, United States
Chan, Andy, Milpitas, CA, United States
PA Openwave Systems Inc., Redwood City, CA, United States (U.S.
corporation)
PI US 6230190 B1 20010508
AI US 1998-169360 19981009 (9)
DT Utility
FS Granted
LN.CNT 705
INCL INCLM: 709/213.000
INCLS: 709/216.000
NCL NCLM: 709/213.000
NCLS: 709/216.000
IC [7]
ICM: G06F015-167
EXF 709/213; 709/214; 709/216; 709/200; 709/203; 709/229

L14 ANSWER 3 OF 4 USPATFULL
AN 2000:62785 USPATFULL
TI Context manager and method for a virtual sales and service center
IN Smith, Jim, Charlotte, NC, United States
Adamczyk, Jim, Chicago, IL, United States
Pesavento, John, Charlotte, NC, United States
PA Andersen Consulting LLP, Chicago, IL, United States (U.S. corporation)
PI US 6064973 20000516
AI US 1998-62492 19980417 (9)
DT Utility
FS Granted
LN.CNT 1092
INCL INCLM: 705/007.000

```
       INCLS: 705/008.000; 705/009.000; 379/220.000
NCL    NCLM:  705/007.000
       NCLS:  379/220.000; 705/008.000; 705/009.000
IC     [7]
       ICM: G06F015-21
       ICS: H04M001-69
EXF    705/7; 705/8; 705/9; 705/11; 705/26; 379/220

L14  ANSWER 4 OF 4  USPATFULL
AN     2000:42166  USPATFULL
TI     Scalable distributed network controller
IN     Ford, Christopher, Princeton, NJ, United States
       Lakshmanan, Mahadevan, Voorhees, NJ, United States
       Scalf, Robert Scott, Mount Holly, NJ, United States
PA     Merrill Lynch & Co. Inc., New York, NY, United States (U.S.
corporation)
PI     US 6047324              20000404
AI     US 1998-19233           19980205 (9)
DT     Utility
FS     Granted
LN.CNT 666
INCL   INCLM: 709/227.000
       INCLS: 709/230.000
NCL    NCLM:  709/227.000
       NCLS:  709/230.000
IC     [7]
       ICM: G06F015-10
EXF    709/300; 709/302; 709/304; 709/227; 709/230; 707/1; 707/10


=> s reliabilit? or availabilit?

        168303 RELIABILIT?
         86141 AVAILABILIT?
L17     245955 RELIABILIT? OR AVAILABILIT?

=> s l14 and l17

L18           4 L14 AND L17

=> d 1-4 fp
```

L25 ANSWER 3 OF 3 USPATFULL
TI Network component **server**
AB A component **server** architecture is provided that enables consumer nodes of a computer network to interact with heterogeneous software components and services distributed. . . components offered in response to requests from applications executing on the consumer nodes. The architecture is implemented on a component **server** node of the network that is configured to communicate with the consumer, i.e., **client**, nodes in **client-server** computing arrangements. That is, the component registry of the component **server** node responds to a consumer application request by locating a heterogeneous component for the consumer. The registry offers this component. . .
SUMM . . . (CLSIDs) to the file system location of a binary object and to type library information about the interface(s) of the **server** component. This provides a framework to make individual pieces of software available, but it is limited to COM objects and. . .
SUMM . . . access to the registration information. While COM components may be stored on computers that are connected to a LAN and **clients** of these components may also reside on the LAN, COM does not take full advantage of the capabilities of LAN. . .
SUMM Distributed COM (**DCOM**) adds remote execution and distributed naming to COM. **DCOM** locates components using the same mechanisms defined in COM (the location can be supplied by the **client** in a local Windows registry). In addition **DCOM** plans to use the NT 5.0 active directory where references to objects are distributed with the naming system. Once **DCOM** has located a component on the network it issues a remote procedure call to the destination system to access remote objects. **DCOM** improves upon COM, but **DCOM** still has limitations in the area of interoperation with heterogeneous object models, and in the use of common LAN administration. . .
SUMM . . . find it. In addition, Java applets and COM objects do not work together without custom written browsers and static object **wrappers** which perform the necessary interface transformations or adaptations.
SUMM . . . the new service with the implementation repository either directly or indirectly via a naming service or Trader service. Thereafter, a **client** on the network accesses the implementation repository either directly (static reference to known physical objects), or via naming or trading. . .
SUMM . . . repository which indexes the CORBA implementation repository. The object reference to the service offer is then passed back to the **client** so that the software objects can bind via the CORBA binding service.
SUMM . . . The CORBA object model addresses the need for interoperation between object models by requiring all non-CORBA objects to wear CORBA **wrappers**, or "jackets". To a CORBA system, all objects may interoperate as long as they "look" like CORBA objects. Accordingly, CORBA. . .
SUMM The present invention addresses these issues and, in particular, is directed to a component **server** architecture that provides an infrastructure for administering, integrating, exposing and managing distributed heterogeneous components and services. More specifically,

the present. . .

SUMM . . . yet another object of the present invention to provide such a system which takes advantage of the capabilities of Internet **servers**.

SUMM The invention comprises a network component **server** architecture that manages and enables consumer nodes of a computer network to interact with heterogeneous software components, hardware devices, data. . .

SUMM In the illustrative embodiment, the inventive architecture is implemented on a component **server** node of the network that is configured to communicate with the consumer, i.e., **client**, nodes in **client-server** computing arrangements. That is, the component registry CMS of the network component **server** node responds to a consumer application request by locating a heterogeneous component for the consumer; according to an aspect of. . .

DETD . . . 104, 106, 108, 110 and 120 connected to a plurality of nodes that may be advantageously used with the component **server** architecture of the present invention. The nodes are typically computers such as local and/or remote mobile systems embodied as personal computers (PC), laptops, workstations, mainframes, minicomputers, network computers (NC), personal digital assistants (PDA), Java Stations, file **servers** and/or application **servers** and network devices. Communication among the nodes is typically effected by exchanging discrete data packets or frames over network signal. . .

DETD The component **server** architecture of the present invention may be configured to reside on any node having sufficient data storage, network connectivity and processing capability. In the illustrative embodiment, however, the architecture is preferably implemented on a component **server** node configured to provide services in response to requests from applications executing on **client**, i.e., consumer, nodes of the network. The component **server** may comprise a single instance or multiple instantiations configured to operate collectively or independently on a node, while the consumers may reside on any node in the network including the same node as the component **server**. These consumer nodes may include non-traditional network nodes such as embedded software devices, hardware devices, printers, routers, modems, databases, database. . .

DETD Network Component **Server** Architecture

DETD According to the invention, the component **server** architecture described herein provides a framework for enabling distributed interaction between a consumer application and heterogeneous software components and services.. . .

DETD If the application 210 is an existing or "Network Component **Server**-unaware" application, then components are preferably accessed via the NCO model 220 which allows backward compatibility through bind services with the. . .

DETD Component requests to the network component **server** may be redirected to the CMS 280 via paths 5 and 22 or called directly via paths 3 and 23.. . .

DETD The component **server** architecture provides redirection processes that allow existing components to interact with the CMS without requiring rewrite of the components' software.. . . single familiar object model using familiar tools, programming languages and interfaces that require little or no knowledge of the component **server**. Through redirection, legacy applications can be exposed as components and managed transparently.

DETD . . . separate from the offer repository 420, object factory repository 430 and the interface adapter repository 440 so that multiple CMS **servers** may share common component definitions. Separation

of the description repository 410 from the other repositories of the component registry 400. . .

DETD . . . attributes and relationships between objects in NDS. The NDS schema may be further modified to allow representation of multiple component **servers** and type repositories at any node in the NDS tree. In the illustrative embodiment, three additional NDS container objects and. . . to the NDS schema as illustrated in FIG. 4B. The container objects include a description repository container 450, a component **server** container 460 and a description container 470, whereas the leaf objects include type objects 452, offer objects 472, implementation objects. . .

DETD . . . and the description repository container 450 to specify configuration information and persistent data for the description repository 410. The component **server** containers 460 hold a desired number of the description type containers 470 which aid in the administration, look-up and organization of implementations and offers. The component **server** containers 460 also have additional attributes and properties to specify configuration information and persistent data for the named component **server**.

DETD The component **server objects** added to the NDS Schema and added to the NDS tree inherit all of the functionality implemented in NDS and. . .

DETD . . . be enforced to limit the number of connections to given services, and to maximize utilization of the CPU. Multiple hardware **servers** can be populated with services as requests are received.

DETD . . . object models. The CMS infrastructure provides information to enable static object model bridging, where hard-coded (program implementation time) interfaces or **wrapper** code is part of the component implementation. Dynamic generation of object bridges requires that a super-set of interface information be. . .

DETD . . . 1. The component registry 620 then provides the appropriate component to the requested application via path 2. If a static **wrapper** implementation exists for the object model of the application, a static proxy code is supplied to the application. The proxy code generates a **wrapper** object 608 for providing a one-to-one bridge via path 3 that is both transparent to the consuming application and is. . . 620 to build proxy code which binds with the remote object or service 630. This method is slower than the object-**wrapper** approach, but it is more flexible in that it provides various object bridges (i.e., bridging from many object-models to many-object. . .

DETD . . . as the look-up and binding point of control. Stated another way, the CMS uses NDS to provide user authentication and **client** identity. The CMS can control who provides components and services by specifying who can add to the object factory and. . .

DETD . . . control of the system, (up to operations on whole types of components, complete groups and organizations, and complete Component Management **Servers**).

DETD As illustrated in FIG. 8B, administrators can browse the component **server** tree 860 for description type containers 870, and associated offer components 872, implementation components 874, and interface components 876 of. . .

DETD . . . NDS loosely replicates the information throughout the entire company XYZ. If a new service implementation is added to the component **server** in the New York 9A partition for example, this service is replicated transparently across the company to the other partitions. . .

CLM What is claimed is:

1. A component **server** for integrating, exposing and managing distributed components residing on a computer network, comprising: a component management service for controlling requests. . .

2. A component **server** according to claim 1, wherein said component management service comprises software for managing the components and services across a plurality. . .

3. A component **server** according to claim 1, wherein said

component registry comprises information for registered components and services so that said component management. . .
4. A component **server** according to claim 1, wherein said component management service further controls authenticated access for registering, discovering and binding of offered. . .
5. A component **server** according to claim 1, wherein said component management service further provides a single point of control for managing hardware and. . .
6. A component **server** according to claim 1, wherein said component management service further provides a single point of control for managing information that. . .
7. A component **server** according to claim 1, wherein said component management service further provides a single point of control for distributing components and. . .
8. A component **server** according to claim 1, wherein said component management service further provides a single point of control for licensing components and. . .
9. A component **server** according to claim 1, wherein said component management service further provides a single point of control for re-directing requested components. . .
10. A component **server** according to claim 1, wherein said component management service further provides a single point of control for bridging object models. . .
11. A component **server** according to claim 1, wherein the component registry and the component management service operate with CORBA, COM and Java RMI. . .
12. A component **server** according to claim 1, wherein the framework for said component registry is based upon NetWare Directory Services (NDS).

13. A component **server** according to claim 12, wherein NDS provides authentication, replication, distribution, and management for the offered and requested components.

14. A component **server** according to claim 12, wherein NDS provides a persistent storage and security access control for said component registry.

15. A component **server** according to claim 1, wherein the framework for said component registry is based upon LDAP, flat file, or ODBC/JDBC databases.
. . .
16. A component **server** according to claim 1, wherein said component management service comprises an interface adapter for transforming the offered component to interface. . .
17. A component **server** according to claim 1, wherein said component registry functions as a central repository for caching of components and services to. . .
18. A component **server** according to claim 17, wherein said central repository permits mobile computing by the requested components and services.

19. A component **server** according to claim 1, wherein said component management service performs load balancing for the requested components and services.

L37 ANSWER 11 OF 17 USPATFULL
United States Patent Patent Number: 6088717
Date of Patent: 11 Jul 2000

Computer-based communication system and method using metadata defining a control-structure

Inventor(s): Reed, Drummond Shattuck, Seattle, WA, United States
Heymann, Peter Earnshaw, Seattle, WA, United States
Mushero, Steven Mark, Seattle, WA, United States
Jones, Kevin Benard, Seattle, WA, United States
Oberlander, Jeffrey Todd, Seattle, WA, United States
Assignee: OneName Corporation, Seattle, WA, United States (U.S. corporation)
Appl. No.: 98-143888
Filed: 31 Aug 1998

Publication Details

PATENT INFORMATION: US 6088717 11 Jul 2000

Related U.S. Application Data

Continuation of Ser. No. US 1996-722314, filed on 27 Sep 1996, now patented, Pat. No. US 5862325, Pat. No. 5862325
which is a continuation-in-part of Ser. No. US 1996-609115, filed on 29 Feb 1996

Int. Cl. G06F015-16
Issue U.S. Cl. 709/201.000; 709/212.000; 709/227.000; 709/229.000; 709/242.000; 709/244.000; 707/010.000; 707/104.000; 707/203.000; 707/204.000
Current U.S. Cl. 709/201.000; 707/010.000; 707/104.100; 707/203.000; 707/204.000; 709/212.000; 709/227.000; 709/229.000; 709/242.000; 709/244.000
Field of Search 709/200-203; 709/212; 709/216-219; 709/227-229; 709/232; 709/242-244; 707/1; 707/9-10; 707/100-104; 707/200-204

Reference Cited

PATENT DOCUMENTS

Patent

L37 ANSWER 5 OF 17 USPATFULL
United States Patent Patent Number: 6189046 B1
Date of Patent: 13 Feb 2001

Mechanism and method for merging cached location information in a distributed object environment

Inventor(s): Moore, Keith E., Santa Clara, CA, United States
Kirshenbaum, Evan, Mountain View, CA, United States
Assignee: Hewlett-Packard Company, Palo Alto, CA, United States (U.S. corporation)
Appl. No.: 97-828027
Filed: 27 Mar 1997

Publication Details

PATENT INFORMATION: US 6189046 B1 13 Feb 2001

Int. Cl. G06F009-54
Issue U.S. Cl. 709/315.000; 709/217.000
Current U.S. Cl. 709/315.000; 709/217.000
Field of Search 395/683; 709/304; 709/303; 709/315; 709/330; 709/310; 709/217; 709/228

Reference Cited

PATENT DOCUMENTS

Patent Number	Date	Class	Inventor
US 5136716	Aug 1992	395/800.000	Harvey et al.
US 5291593	Mar 1994	395/600.000	Abraham et al.
US 5539909	Jul 1996	395/700.000	Tanaka et al.
US 5724588	Mar 1998	395/684.000	Hill et al.
US 5737607	Apr 1998	395/701.000	Hamilton et al.
US 5758186	May 1998	395/831.000	Hamilton et al.
US 5802590	Sep 1998	711/164.000	Draves
US 5892910	Apr 1999	395/200.470	Safadi

OTHER PUBLICATIONS

(Chappell) David Chappell. "Understanding ActiveX and OLE" p. 51-52, Sep. 17, 1996.

Birrell, Andrew et al. "Network Objects" p. 219-221, Dec. 1993.

BNR Europe Limited. "OMG Object Request Broker 2.0 Interoperability and Initialisation RFP Response". p. 1-29, Mar. 4, 1994.

ICL submission to the OMG Object Request Broker 2.0. "ORB Interoperability"., p. 7-14, Mar. 7, 1994.

Art Unit - 275
Primary Examiner - Oberley, Alvin E.
Assistant Examiner - Bullock, Jr., Lewis A.

---

13 Claim(s), 18 Drawing Figure(s), 14 Drawing Page(s)

ABSTRACT

In a method of operating a computer system having a plurality of processes, creating a plurality of object references, each object reference corresponding to a target object. The processes optionally executing on a plurality of computers connected by a network. For each object reference, creating a table of binding information hints. The table is indexed by a particular transport protocol and each entry in the table of binding information hints includes information to be used to attempt to establish a connection from the process to
the target object using the indexing transport protocol. Merging the tables of binding information hints upon receiving an object reference.

Other systems and methods are disclosed.

L42 ANSWER 5 OF 16 USPATFULL
United States Patent Patent Number: 6212556 B1
Date of Patent: 3 Apr 2001

Configurable value-added network (VAN) switching

Inventor(s): Arunachalam, Lakshmi, Menlo Park, CA, United States
Assignee: WebXchange, Inc., Menlo Park, CA, United States (U.S. corporation)
Appl. No.: 99-296207
Filed: 21 Apr 1999

Publication Details

PATENT INFORMATION: US 6212556 B1 3 Apr 2001

Related U.S. Application Data

Continuation-in-part of Ser. No. US 1999-879958, filed on 20 Jun 1999, now patented, Pat. No. US 5987500, Pat. No. 5987500
Division of Ser. No. US 1996-700726, filed on 5 Aug 1996, now patented, Pat. No. US 5778178, Pat. No. 5778178

Priority Data

US 1995-6634 13 Nov 1995 (60)

Int. Cl. G06F013-00
Issue U.S. Cl. 709/219.000; 709/223.000; 709/225.000; 709/227.000
Current U.S. Cl. **709/219.000; 709/223.000; 709/225.000; 709/227.000**
Field of Search 709/202; 709/203; 709/217; 709/218; 709/219; 709/223; 709/224; 709/225; 709/227; 709/230; 709/238; 709/250; 709/313; 709/328; 709/329

Reference Cited

PATENT DOCUMENTS

Patent Number	Date	Class	Inventor
US 5148474	Sep 1992	379/111.000	Haralambopoulos et al.
US 5537464	Jul 1996	379/114.000	Lewis et al.
US 5557780	Sep 1996	395/500.480	Edwards et al.
US 5828666	Oct 1998	370/389.000	Focsaneanu et al.
US 5892821	Apr 1999	379/220.000	Turner

OTHER PUBLICATIONS

Davidson, Andrew, Coding with HTML forms: HTML goes interactive. (hpertext markup language) (Tutorial); Dr. Dobb's Journal; vol. 20, No. 6, Jun. 1995; 19 pages.

Art Unit - 214
Primary Examiner - Vu, Viet D.

Attorney, Agent or Firm - Blakely, Sokoloff, Taylor & Zafman, LLP

---

30 Claim(s), 30 Drawing Figure(s), 29 Drawing Page(s)

ABSTRACT

The present invention provides a method and apparatus for providing real-time, two-way transactional capabilities on the network. Specifically, one embodiment
of the present invention discloses a configurable value-added network switch for enabling real-time transactions on the network. The configurable value added network switch comprises means for switching to a transactional application in response to a user specification from a network application, means for transmitting a transaction request from the transactional application, and means for processing the transaction request. Additionally, a method for enabling object routing is disclosed, comprising the steps of creating a virtual information store containing information entries and attributes associating each of the information entries and the attributes with an object identity, and assigning a unique network address to each of the object identities. Finally, a method is disclosed for enabling service management of the value-added network service, to perform OAM&P functions on the services network.

L42 ANSWER 15 OF 16 USPATFULL
United States Patent Patent Number: 6041365
Date of Patent: 21 Mar 2000

Apparatus and method for high performance remote application gateway ***servers***

Inventor(s): Kleinerman, Aurel, 307 Waverly St. #5, Menlo Park, CA, United States 94025
Appl. No.: 97-885141
Filed: 30 Jun 1997

Publication Details

PATENT INFORMATION: US 6041365 21 Mar 2000

Related U.S. Application Data

Continuation-in-part of Ser. No. US 1995-542863, filed on 13 Oct 1995, now patented, Pat. No. US 5734871, Pat. No. 5734871
which is a continuation of Ser. No. US 1995-406638, filed on 20 Mar 1995, now
abandoned
which is a continuation of Ser. No. US 1994-261764, filed on 17 Jun 1994, now
abandoned
which is a continuation of Ser. No. US 1993-89947, filed on 12 Jul 1993, now abandoned
which is a continuation of Ser. No. US 1990-549889, filed on 9 Jul 1990, now patented, Pat. No. US 5228137, Pat. No. 5228137
which is a continuation-in-part of Ser. No. US 1988-145692, filed on 15 Jan 1988, now abandoned
which is a continuation of Ser. No. US 1985-792424, filed on 29 Oct 1985, now
abandoned

Int. Cl. G06F015-00
Issue U.S. Cl. 709/302.000
Current U.S. Cl. **709/328.000**
Field of Search 707/103; 395/200.15; 395/500; 709/228; 709/235; 709/229;
709/231; 709/226; 709/302

Reference Cited

PATENT DOCUMENTS

Patent Number	Date	Class	Inventor
US 5680551	Oct 1997	395/200.560	Martino, II
US 5745901	Apr 1998	707/103.000	Entner et al.
US 5790809	Aug 1998	395/200.580	Holmes
US 5822521	Oct 1998	395/200.600	Gartner et al.
US 5828842	Oct 1998	395/200.530	Sugauchi et al.

Art Unit - 274